Initial State Training Procedure Improves Dynamic Recurrent Networks With Time-Dependent Weights

Lutz Leistritz, Miroslaw Galicki, Herbert Witte, and Eberhard Kochs

***Abstract*—The problem of learning multiple continuous trajectories by means of recurrent neural networks with (in general) time-varying weights is addressed in this study. The learning process is transformed into an optimal control framework where both the weights and the initial network state to be found are treated as controls. For such a task, a new learning algorithm is proposed which is based on a variational formulation of Pontryagin's maximum principle. The convergence of this algorithm, under reasonable assumptions, is also investigated. Numerical examples of learning nontrivial two-class problems are presented which demonstrate the efficiency of the approach proposed.**

***Index Terms*—Dynamic neural networks, initial network states, optimal control, trajectory learning.**

I. Introduction

Artificial neural networks with recurrent connections and dynamic processing elements (neurons) have recently been applied to interesting diverse dynamic tasks including associative memories, identification and/or control of dynamic systems, forecasting, generation of pattern sequences, and spatio-temporal pattern classification. Theoretical works by several researchers [1]–[3] have proved that the recurrent networks are universal approximators of dynamic systems. A common problem for recurrent neural networks in all of the foregoing tasks is to find an underlying mapping which correspondingly relates their input–output spaces based on a given learning set. It consists (usually) of a finite number of learning pairs which present time-varying sequences (trajectories). Consequently, the reconstruction of the aforementioned mapping becomes equivalent to learning the corresponding trajectories using a neural network. A number of approaches may be distinguished in this context [2]–[18]. Pearlmutter [6], [7], Werbos [8], Toomarian and Barchen [9] and Draye *et al.* [10] have developed an algorithm, known as backpropagation through time (BPTT), using gradient-descent-based methods. It learns time-varying external inputs and produces either desired temporal behaviors over a bounded time interval or trains nonfixed-point attractors. Willims and Zipser [11], Meert and Ludik [12] constructed a gradient descent learning rule which they called real-time recurrent learning (RTRR) and which can deal with time sequences of arbitrary length. A somewhat different architecture than that of [11] was proposed in [13], where the authors showed a stochastic convergence of the learning algorithm for their recurrent network. A comprehensive overview of gradient descent learning algorithms for recurrent neural networks has been given in [14]. A stochastic search method based on an adaptive simulated annealing algorithm was used by Cohen *et al.* [15] to efficiently train recurrent neural networks with time delays (TDRNNs). An effort was made in the above investigation to implement several benchmark tasks using minimum-size networks. Recurrent multilayer perceptron networks (RMLPs) were investigated in [16]. They are constructed from a multilayer perceptron architecture and then by adding delayed connections among the neighboring neurons of the hidden layers. However, only empirical evidence indicates that, as a result of delayed recurrent connections, RMLPs emulate a large class of nonlinear dynamic systems. Based on the minimization of an error function, constant weights in a time interval of network evolution were obtained after carrying out the learning process in [3]–[16]. Neural networks with time-varying weights were proposed in the works [17], [18]. Inserting time-dependent weights results in an overparameterization of such networks. However, this property gives us the possibility to define additional new tasks (by maintaining the conventional ones), thus reducing (or limiting) network parameter redundancy. The networks with time-varying weights have several advantages such as smaller structures and a broader class of trainable trajectories when compared with conventional (constant weights) networks. The influence of a negative initial network weights distribution on the learning speed of recurrent neural networks was discussed from the statistical point of view in [19]. Analyzing the behavior of the mean activity of the network versus time, the authors have tried to explain the influence of an inhibitory network weight initialization.

In all of the above studies, a fixed (i.e., arbitrarily specified by the user) initial network state was assumed. However, specification (performed usually randomly) of a poor initial state may considerably slow down the convergence rate of a learning procedure or even lead to difficulties in its convergence. In order to overcome these drawbacks, both the weights and the initial state of the network are subject to the learning process in this work. The learning process has been formulated as an optimal control problem, where the weights and initial network state are treated as unknown controls (with control and state dependent constraints). In addition, a multiple-class problem addressed in this study enforces natural boundary state equality constraints on the network trajectories. In order to solve this, a new algorithm is proposed herein which generalizes that given in [18].

Manuscript received May 22, 2000; revised April 20, 2001. This work was supported by the BMBF (Clinical oriented Neurosciences, 01ZZ9602) and by the Deutsche Forschungsgemeinschaft (DFG, WI 1166/4-1).

L. Leistritz and H. Witte are with the Institute of Medical Statistics, Computer Sciences and Documentation, Friedrich Schiller University, Jena, Germany.

M. Galicki is with the Institute of Medical Statistics, Computer Sciences and Documentation, Friedrich Schiller University, Jena, Germany. He is also with the Institute of Organization and Management, Technical University of Zielona Gora, Zielona Gora, Poland.

E. Kochs is with the Institute of Medical Statistics, Computer Sciences and Documentation, Friedrich Schiller University, Jena, Germany. He is also with the Department of Anesthesiology, Technical University Munich, Munich, Germany.

Publisher Item Identifier S 1045-9227(01)09519-4.

It is based on a variational formulation of the Pontryagin maximum principle [18], [20]. Using reasonable assumptions, a convergence of the new algorithm to an optimal solution is also shown.

The outline of this paper is as follows. A learning task of multiple continuous trajectories is formulated in Section II. Based on a variational formulation of Pontryagin's maximum principle, the new learning algorithm is proposed in Section III. Numerical examples—simulated and real-life ones—of learning two-class problems are presented in Section IV. Finally, some concluding remarks are made in Section V.

II. The Problem of Learning Multiple Trajectories

Let us consider a dynamic neural network whose state is determined by the differential and algebraic equations of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}), \qquad \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ is a state vector of the network. n denotes the number of all neurons. $\boldsymbol{w} = \left(w_{(i-1)n+j}\right)_{1 \le i,j \le n} \in \mathbb{R}^{n^2}$ stands for the weight vector whose coordinates may depend, in general, on time ($w_{(i-1)n+j}$ represents the weight connection out of neuron i to neuron j) and $\boldsymbol{u} = \boldsymbol{u}(t) = (u_1, \ldots, u_n)^T$ stands for an external input into the network. The mapping $\boldsymbol{f} = (f_1, \ldots, f_n)^T$ is given by

$$f_i(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) = -x_i + h\left(\sum_{j=1}^{n} w_{(i-1)n+j} x_j\right) + u_i, i = 1, \ldots, n \tag{2}$$

with a strictly increasing and differentiable neural activation function h. The network evolution time T is finite. $\boldsymbol{x}(0) = \boldsymbol{x}_0 = (x_{1,0}, \ldots, x_{n,0})^T$ denotes an initial state of the network. The network output is defined by a differentiable mapping $\boldsymbol{g} : \mathbb{R}^n \longrightarrow \mathbb{R}^k$, $n \ge k$. It usually takes on the following form:

$$\boldsymbol{g}(\boldsymbol{x}) = (x_1, \ldots, x_k)^T \tag{3}$$

such that the states of neurons $1, \ldots, k$ represent the outputs of the network. For the sake of simplicity of further considerations, time constants of the neurons, as well as their synaptic delays have been omitted herein.

The natural requirement when learning multiple classes is that the initial state of the network should be the same for all output trajectories. So far, all previous publications have assumed that the initial state is known (i.e., arbitrarily specified by the user). However, specification of a poor initial state may considerably slow down a convergence rate of a learning process or even lead to difficulties in its convergence. In order to eliminate this drawback, the initial state of the network is assumed herein to be not specified but it will be attained during a training procedure. Hence, the task of learning multiple trajectories is now to modify the weight vector $\boldsymbol{w}$ and the initial state $\boldsymbol{x}_0$ so that the output trajectories $\boldsymbol{y}^c(t)$ corresponding to the external inputs $\boldsymbol{u}^c(t) = (u_1^c(t), \ldots, u_n^c(t))^T$ follow prescribed differentiable target trajectories $\boldsymbol{r}^c(t) \in \mathbb{R}^k$, for all $c = 1, \ldots, C$. $C \ge 1$ denotes a given number of training pairs. Without any loss of generality, we will further limit ourselves by considering an error function of the form

$$\begin{aligned} E(\boldsymbol{x}_0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^C, \boldsymbol{w}, \boldsymbol{u}^1, \ldots, \boldsymbol{u}^C) &= \frac{1}{2}\sum_{c=1}^{C} \|\boldsymbol{y}^c - \boldsymbol{r}^c\|^2 \\ &= \frac{1}{2}\sum_{c=1}^{C} \|\boldsymbol{g}(\boldsymbol{x}^c) - \boldsymbol{r}^c\|^2 \end{aligned} \tag{4}$$

which reflects the temporal behavior of the network and is the form mostly found in the literature. More general cost functions are also possible (see [18]). More formally, the learning task may be expressed, as follows:

Minimize:

$$J(\boldsymbol{w}, \boldsymbol{x}_0) = \int_0^T E dt \tag{5}$$

subject to the constraints

$$\dot{\boldsymbol{x}}^c = \boldsymbol{f}(\boldsymbol{x}^c, \boldsymbol{w}, \boldsymbol{u}^c) \tag{6}$$

and

$$\boldsymbol{x}^c(0) = \boldsymbol{x}_0 = (x_{1,0}, \ldots, x_{n,0})^T \tag{7}$$

for all

$$c = 1, \ldots, C.$$

Some (or all) coordinates of the initial state $\boldsymbol{x}_0$ are assumed to be unknown. In order to fully specify differential equation (1), they should be determined during the training process of the network. As it was argued in [18], time-dependent weights usually lead to a better overall performance of the neural network. In such a case they may be treated as controls. Thus, the expressions (5)–(7) formulate the learning task as an optimal control problem. The next section will present an approach based on the variational formulation of Pontryagin's maximum principle which renders it possible to solve the aforementioned optimization problem.

III. Learning Continuous Trajectories

The application of the variational formulation of the Pontryagin maximum principle to the problem specified by dependencies (1), (5)–(7) requires a knowledge of the initial weights $\boldsymbol{w}_{[0]} = \boldsymbol{w}_{[0]}(t)$, $t \in [0, T]$ and the initial state vector $\boldsymbol{x}_0$. The initial trajectory $\boldsymbol{x}_{[0]}(t) = (\boldsymbol{x}_{[0]}^1(t), \ldots, \boldsymbol{x}_{[0]}^C(t))^T$ is the result of solving differential equation (6) for $\boldsymbol{u}^c$, $c = 1, \ldots, C$, with respect to $\boldsymbol{w}_{[0]}$. Moreover, the quantities $\boldsymbol{w}_{[0]}$ and $\boldsymbol{x}_0$ are assumed not to minimize performance index (5) (see [18] and [21] for the details of the variational formulation of Pontryagin's maximum principle). The use of this (nonclassical) formulation requires the incrementation of functional (5). Therefore it is assumed that the weight vector $\boldsymbol{w}_{[0]} = \boldsymbol{w}_{[0]}(t)$ and the initial state $\boldsymbol{x}_0$ are perturbed by small variations $\delta\boldsymbol{w} = \delta\boldsymbol{w}(t) \in \mathbb{R}^{n^2}$ and $\delta\boldsymbol{x}_0 = (\delta x_{1,0}, \ldots, \delta x_{n,0})^T$, where $\|\delta\boldsymbol{w}\|_\infty \le \rho$ and $\|\delta\boldsymbol{x}_0\|_\infty \le \xi$ ($\|\cdot\|_\infty$ denotes the maximum norm). Thereby, the constants

ρ and ξ are given small numbers ensuring the correctness of the presented method. From the practical point of view, values within the range of $[10^{-4}, 10^{-3}]$ have been approved.

Let $\boldsymbol{f_w}$, $\boldsymbol{f_x}$ and $E_{\boldsymbol{x}}$ be the following differentials:

$$\boldsymbol{f_w}(c,t) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\right)_{\substack{\boldsymbol{x}=\boldsymbol{x}^c_{(0)}\\ \boldsymbol{w}=\boldsymbol{w}_{|0|}\\ \boldsymbol{x}^c(0)=\boldsymbol{x}_0}}$$

$$\boldsymbol{f_x}(c,t) = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{\substack{\boldsymbol{x}=\boldsymbol{x}^c_{(0)}\\ \boldsymbol{w}=\boldsymbol{w}_{|0|}\\ \boldsymbol{x}^c(0)=\boldsymbol{x}_0}}$$

$$E_{\boldsymbol{x}}(c,t) = \left(\frac{\partial E}{\partial \boldsymbol{x}}\right)_{\substack{\boldsymbol{x}=\boldsymbol{x}^c_{(0)}\\ \boldsymbol{w}=\boldsymbol{w}_{|0|}\\ \boldsymbol{x}^c(0)=\boldsymbol{x}_0}}.$$

Then, neglecting the higher order terms $\boldsymbol{o}(\delta\boldsymbol{w}, \delta\boldsymbol{x}_0)$, the value of functional (5) for the perturbed weights $\boldsymbol{w}_{[0]} + \delta\boldsymbol{w}$ and initial state $\boldsymbol{x}_0 + \delta\boldsymbol{x}_0$ may thus be expressed in the form

$$J(\boldsymbol{w}_{[0]} + \delta\boldsymbol{w}, \boldsymbol{x}_0 + \delta\boldsymbol{x}_0) = J(\boldsymbol{w}_{[0]}, \boldsymbol{x}_0) + \delta J(\boldsymbol{w}_{[0]}, \boldsymbol{x}_0, \delta\boldsymbol{w}, \delta\boldsymbol{x}_0) \tag{8}$$

where

$$\delta J(\boldsymbol{w}_{[0]}, \boldsymbol{x}_0, \delta\boldsymbol{w}, \delta\boldsymbol{x}_0) = \sum_{c=1}^{C}\left(\int_0^T \langle(\boldsymbol{f_w}(c,t))^T \cdot \boldsymbol{\Psi}^c, \delta\boldsymbol{w}\rangle\, dt + \langle\boldsymbol{\Psi}^c(0), \delta\boldsymbol{x}_0\rangle\right) \tag{9}$$

is the Frechet differential of the functional J ($\langle\cdot,\cdot\rangle$ denotes the scalar product of vectors). $\boldsymbol{\Psi}^c$ denote the conjugate mappings, which are determined by the solution of the Cauchy problem

$$\dot{\boldsymbol{\Psi}}^c + \boldsymbol{f}^T_{\boldsymbol{x}}(c,t)\cdot\boldsymbol{\Psi}^c = -E_{\boldsymbol{x}}(c,t), \qquad \boldsymbol{\Psi}^c(T) = \boldsymbol{0}. \tag{10}$$

For properly selected variations $\delta\boldsymbol{w}$ and $\delta\boldsymbol{x}_0$, the Frechet differential of functional (5) approximates the increment of this functional with any desired accuracy. The variational formulation of Pontryagin's maximum principle takes the following form:

$$\begin{aligned} &J(\boldsymbol{w}_{[0]} + \delta\boldsymbol{w}^*\, h, \boldsymbol{x}_0 + \delta\boldsymbol{x}^*_0)\\ &= J(\boldsymbol{w}_{[0]}, \boldsymbol{x}_0)\\ &\quad + \min_{(\delta\boldsymbol{w}, \delta\boldsymbol{x}_0)}\{\delta J(\boldsymbol{w}_{[0]}, \boldsymbol{x}_0, \delta\boldsymbol{w}, \delta\boldsymbol{x}_0)\} \end{aligned} \tag{11}$$

subject to the constraints

$$\begin{aligned} \|\delta\boldsymbol{w}^*\|_\infty \le& \rho, \quad \|\delta\boldsymbol{x}^*_0\|_\infty \le \xi,\\ \boldsymbol{x}^c(0) =& \boldsymbol{x}_0 + \delta\boldsymbol{x}^*_0, \quad c = 1,\ldots,C. \end{aligned} \tag{12}$$

Clearly, the linearized optimal control problem (11) and (12) may be solved analytically with respect to the variations $\delta\boldsymbol{w}^*$ and $\delta\boldsymbol{x}^*_0$.

The term in (9) and (11) which depends on $\delta\boldsymbol{x}_0$ is given by $\sum_{c=1}^{C}\langle\boldsymbol{\Psi}^c(0), \delta\boldsymbol{x}_0\rangle$. Since $\delta\boldsymbol{x}_0$ is independent of c

$$\sum_{c=1}^{C}\langle\boldsymbol{\Psi}^c(0), \delta\boldsymbol{x}_0\rangle = \left\langle\sum_{c=1}^{C}\boldsymbol{\Psi}^c(0), \delta\boldsymbol{x}_0\right\rangle \tag{13}$$

holds. Evidently, the right-hand side is minimized subject to the constraints (12) by

$$\delta\boldsymbol{x}^*_0 = -\xi\cdot\operatorname{sgn}\left(\sum_{c=1}^{C}\boldsymbol{\Psi}^c(0)\right). \tag{14}$$

in the case when all coordinates of $\boldsymbol{x}_0$ are subject to the learning process.

The weight variation $\delta\boldsymbol{w}^*(t)$, $t \in [0, T]$, may be determined using a very efficient numerical procedure that has been proposed in [18]. It is based on a finite-dimensional approximation of $\delta\boldsymbol{w}$ in (11) and (12). More precisely, the weight variation $\delta\boldsymbol{w}$ is approximated by a finite sum of linear combinations of, e.g., Chebyshev's polynomials whose coefficients are subject to the minimization process. If some coordinates of vector $\boldsymbol{x}_0$ are specified (e.g., by the user) then their corresponding variations equal zero. The minimization process (11) and (12) is then repeated for the revised weights $\boldsymbol{w}_{[1]} = \boldsymbol{w}_{[0]} + \delta\boldsymbol{w}^*$ and initial state $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \delta\boldsymbol{x}^*$. A sequence of pairs $(\boldsymbol{w}_{[k]}, \boldsymbol{x}_k)$, $k = 0, 1, \ldots$, may be thus obtained as a result of solving the iterative approximation scheme (11)–(12). Each element of this sequence corresponds to a state trajectory $\boldsymbol{x}_{[k]}$ according to (6). Provided that a limit exists for $(\boldsymbol{w}_{[k]}\colon k = 0, 1, \ldots)$, the sequence $((\boldsymbol{w}_{[k]}, \boldsymbol{x}_k, \boldsymbol{x}_{[k]})\colon k = 0, 1, \ldots)$ converges to an optimal solution (in general local) if the set $\{\boldsymbol{x}_{[k]}\colon k = 0, 1, \ldots\}$ is bounded [18].

In order to compare the influence of the initial network state training, paired computer simulations were performed on nontrivial discrimination tasks in the next section. Thereby, starting from any common set of initial network states, two coupled training trials with optimized or fixed $\boldsymbol{x}_0$ were performed, respectively. The special case without optimization initial states reduces to the algorithm described in [18].

IV. Computer Examples

The purpose of this section is to show, by means of nontrivial examples of discrimination tasks, the efficiency and global nature of the training procedure proposed in Section III. The computations do not aim to find the minimal number of network connections.

A. Benchmark Task

In order to be accurate, the first task was to learn a two-dimensional two-class problem by means of a fully connected network of three neurons

$$\dot{\boldsymbol{x}} = \begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} = \begin{pmatrix}-x_1 + \arctan(\sum_{j=1}^{3} w_j x_j)\\ -x_2 + \arctan(\sum_{j=1}^{3} w_{j+3} x_j)\\ -x_3 + \arctan(\sum_{j=1}^{3} w_{j+6} x_j) + u_3\end{pmatrix}. \tag{15}$$

From the theoretical point of view, a network with two neurons ($n \ge \sqrt{C\cdot k}$) would be sufficient to solve this task [18]. On the other hand, for practical reasons, it is adverse to provide an output neuron with an external input sometimes (especially in cases with noisy inputs). So, we prefer networks with separated input and output neurons.

In order to reduce the number of parameters which are to be optimized during the training process, the order of basis functions for the approximation of time-varying weights should be

TABLE I
THE NUMBER OF OBSERVED EXPERIMENT OUTCOMES FOR 200 TRIALS

	$K_o < K_f$	$K_o > K_f$
$J_o^* < J_f^*$	53	91
$J_o^* > J_f^*$	40	16

made as small as possible. Unfortunately, to our knowledge there is no general rule to compute this quantity. Generally, the higher the temporal variation of the inputs and outputs the higher the necessary order for a given learning task. Empirical investigations have indicated the the largest order that is necessary to fit the inputs and outputs can be used as an upper limit for the order of basis functions. However, starting with this upper limit a systematic reduction of the number of basis functions seems to be advisable until the training goal can no longer be achieved.

Using this paradigm, time-varying weights described by Chebyshev's polynomials of order six were used for this example. The network output was provided by

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}, \quad t \in [0, 4\pi]. \tag{16}$$

Presented with a constant external input, $u_3^1(t) = 1$, the network is forced to imitate an elliptical trajectory $\boldsymbol{r}^1(t) = (0.5 + 0.45 \cdot \cos(t), 1.0 + 0.9 \cdot \sin(t))^T$; presented with a zero input, $u_3^2(t) = 0$, the network is forced to imitate a "figure eight" trajectory $\boldsymbol{r}^2(t) = (0.5 - 0.45 \cdot \sin(2t), 1.0 + 0.9 \cdot \cos(t))^T$ in the state space.

In order to investigate the influence of training $\boldsymbol{x}_0$, a collection of 200 samples of initial network states and weights were randomly selected according to

$$\begin{aligned} \boldsymbol{X}(0) \sim & \frac{1}{4} \cdot N(0,1)^3 + \frac{1}{2} \cdot \begin{pmatrix} r^1(0) + r^2(0) \\ 0 \end{pmatrix} \\ \boldsymbol{W}_{[0]}(t) = & \boldsymbol{W}_{[0]} \sim \frac{1}{4} \cdot N(0,1)^9 \end{aligned}$$

where $N(0,1)$ denotes the standardized normal distribution. In the computations, all coordinates of vector $\boldsymbol{x}(0)$ were subject to the minimization process in the learning procedure from Section III. The weight variations $\delta \boldsymbol{w}$ were approximated by Chebyshev's polynomials of sixth order.

Starting from the same initial conditions, both the algorithm from [18] and that from Section III (with $\rho = \xi = 10^{-3}$) were run in each experiment until the minimal value of functional (5) was reached. In order to evaluate the performance of both algorithms, the number of iterations K, that were necessary to reach a minimum of (5) and the minimal value of J^*, itself, were taken into account. The results of the computer simulations are presented in Table I. The subscripts o and f are used to denote trials with an optimized or fixed $\boldsymbol{x}_0$, respectively. As is seen, the optimization of the initial network state leads to a significant increase in the number of better local minima (exact binomial test, significance level 0.001). Fig. 1 shows the corresponding trends of the error functional (5). For this example the conditions $K_o > K_f$ and $J_o^* < J_f^*$ hold.

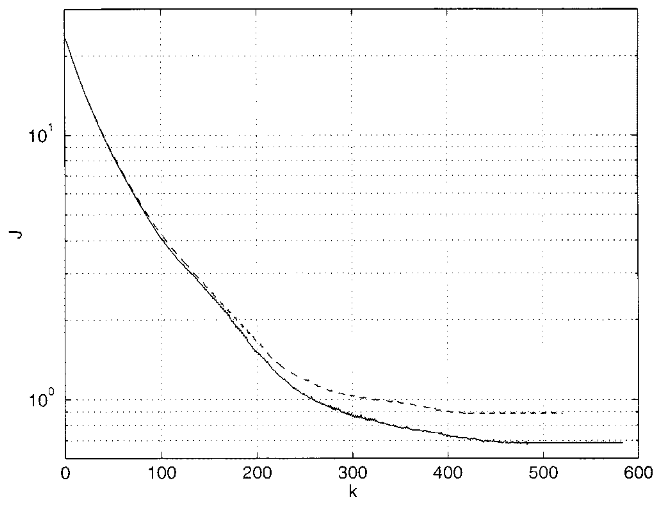

Fig. 1. The trend of the error functional J during the learning process for both algorithms. The solid line corresponds to the improved algorithm whereas the dotted one corresponds to the procedure with fixed $\boldsymbol{x}_0$ (37th experiment).

B. A Real-Life Example

The second task was to learn a one-dimensional two-class problem by means of a fully connected network of four neurons. In fact, the term "two-class" relates to the usage of two different target trajectories, these being associated with 15 different input signals, i.e., $C = 15$.

The patterns used as inputs are 15 middle latency auditory evoked potentials (MLAEP) of patients under anaesthesia, randomly selected from a study described in [22]. Every MLAEP is the average of 512 single sweeps sampled at a rate of 1 kHz. Binaural auditory stimulation was performed at 6.1 Hz with rarefaction clicks using insert earphones. The signals were filtered using an analog 400-Hz low-pass filter and a digital 25-Hz high-pass filter and additionally a three point smoothing was applied. The first 80 ms of the signals were used and the arrangement of the MLAEPs into the categories "nonmover" (eight patients) or "mover" (seven patients) has been carried out according to the patient's reaction to skin incision (see Fig. 2).

The early cortical MLAEP has been shown to be able to indicate intraoperative awareness [23]. It is well known that the analysis of MLAEP-waveforms in the time domain by determination of specific peaks and identification of latencies and amplitudes provides information related to depth of anaesthesia [24]. Unfortunately, at present the required information, e.g., in the form of single parameters, cannot be derived automatically. The identification of these parameters by visual inspection is time consuming and cannot be performed on-line. In addition, due to the mostly noisy signals, the determination of peaks and latencies are subject to individual experience and hence are not unambiguous. From this point of view, an automatic procedure for reproducible parameter calculation and extraction from the AEP-waveforms is highly desirable.

The main purpose of this example is the investigation of the influence of trained initial network states. On the other hand, it can be demonstrated additionally that it is possible to separate two categories of very similar AEP-waveforms (Fig. 2) by means of a fully connected network of minimal size. The network was forced to take the constant state $-1/2$ in the time

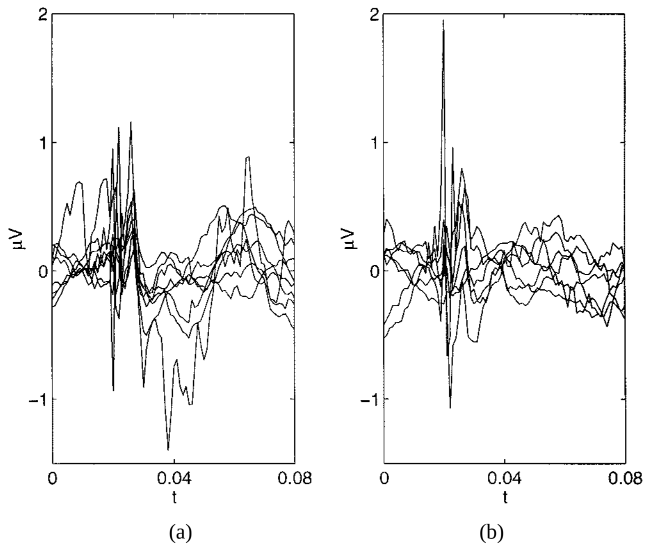

Fig. 2. MLAEPs used as inputs for the network (17). (a) category "nonmover." (b) category "mover."

interval [0.03, 0.08], when a MLAEP from the category "nonmover" was presented. Otherwise, the target was set to 1/2. The network used is defined by

$$\dot{\boldsymbol{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} -x_1+50\cdot\arctan\left(\sum_{j=1}^4 w_j x_j\right) \\ -x_2+50\cdot\arctan\left(\sum_{j=1}^4 w_{j+4} x_j\right) \\ -x_3+50\cdot\arctan\left(\sum_{j=1}^4 w_{j+8} x_j\right) \\ -x_4+50\cdot\left(\arctan\left(\sum_{j=1}^4 w_{j+12} x_j\right)+u_4\right) \end{pmatrix} \quad (17)$$

with time-varying weights described by Chebyshev's polynomials of order 51. The network output is provided by $x_1(t)$. As in the previous section, several training trials were performed with and without optimization of $\boldsymbol{x}_0$. For it 40 samples of initial network states and weights were randomly selected according to

$$\boldsymbol{X}(0) \sim \frac{1}{4}\cdot N(0,1)^4, \quad \boldsymbol{W}_{[0]}(t) = \boldsymbol{W}_{[0]} \sim \frac{1}{4}\cdot N(0,1)^{16}.$$

All coordinates of vector $\boldsymbol{x}(0)$ were subject to the minimization process in the learning procedure from Section III. The weight variations $\delta\boldsymbol{w}$ were approximated by Chebyshev's polynomials of 51st order.

The results of computer simulations are presented in Table II. In 25 of the 40 trials, the optimization of the initial network state leads to a better local minimum of the error functional J.

Finally, the influence of an inhibitory network weight initialization described in [19] could not be observed in both examples. Most probably, this effect is compensated by the time-varying weights.

V. Conclusion

In this study the improved learning algorithm for a neural network with (in general) time-varying weights was presented and tested on discrimination tasks with two classes. The learning procedure has been formulated as an optimal control problem with state dependent constraints. Its convergence to an optimal solution was also discussed. As was mentioned in [18] using neural networks with time-varying weights leads to a significant reduction of both mismatch error and network sizes when compared with those previously reported by [7], [12], [15]. Nevertheless, modification of an initial network state in order to improve the convergence of learning algorithms was not considered before in all aforementioned works although it is important as the computer simulations show. The results presented in this study indicate the need to consider evaluation criteria other than the error functional (5) (e.g., a performance index reflecting generalization error) in order to enhance the performance of dynamic networks. We would like to explore this subject in future works.

TABLE II
THE NUMBER OF OBSERVED EXPERIMENTAL OUTCOMES FOR 40 TRIALS

	$K_o < K_f$	$K_o > K_f$
$J_o^* < J_f^*$	8	17
$J_o^* > J_f^*$	14	1

References

[1] K. I. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Neworks*, vol. 6, pp. 801–806, 1993.
[2] E. A. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. I. Oannou, "High-order neural network structures for identification of dynamical systems," *IEEE Trans. Neural Networks*, vol. 6, pp. 422–431, Mar. 1995.
[3] M. Galicki, L. Leistritz, and H. Witte, "Dynamical multilayer neural networks that learn continuous trajectories," *Pattern Recognition and Image Analysis (Special Issue)*, vol. 9, no. 4, pp. 604–608, 1999.
[4] S. Fabri and V. Kadirkamanathan, "Dynamic structure neural networks for stable adaptive control of nonlinear systems," *IEEE Trans. Neural Networks*, vol. 7, pp. 1151–1166, 1996.
[5] C. C. Ku and K. Y. Lee, "Diagonal neural networks for dynamic systems control," *IEEE Trans. Neural Networks*, vol. 6, pp. 144–156, 1995.
[6] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computat.*, vol. 1, no. 2, pp. 263–269, 1989.
[7] ——, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, pp. 1212–1228, Sept. 1995.
[8] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, 1990.
[9] N. Toomarian and J. Barhen, "Adjoint operators and nonadiabatic algorithms in neural networks," *Applied Math. Lett.*, vol. 4, pp. 69–73, 1991.
[10] J. P. S. Draye, D. A. Pavisic, G. A. Cheron, and G. A. Libert, "Dynamic recurrent neural networks: A dynamical analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 692–706, 1996.
[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[12] K. Meert and J. Ludik, "A multilayer real-time recurrent learning algorithm for improved convergence," in *Proc. Artificial Neural Networks-ICANN'97*, 1997, pp. 505–510.
[13] H. Song, S. M. Kang, and S. W. Lee, "A new recurrent neural network architecture for pattern recognition," in *Proc. ICPR'96, 1996 IEEE*, 1996, pp. 718–722.
[14] P. Baldi, "Gradient descent learning algorithm overview: A general dynamical system perspective," *IEEE Trans. Neural Networks*, vol. 6, pp. 182–195, 1995.
[15] B. Cohen, D. Saad, and E. Marom, "Efficient training of recurrent neural network with time delays," *Neural Networks*, vol. 10, no. 1, pp. 51–59, 1997.
[16] A. G. Parlos, K. T. Chong, and A. F. Atiya, "Application of the recurrent multilayer perceptron in modeling complex process dynamics," *IEEE Trans. Neural Networks*, vol. 5, pp. 255–266, Mar. 1994.

[17] L. Leistritz, M. Galicki, and H. Witte, "Training continuous trajectories by means of dynamic neural networks with time dependent weights," in *Proc. Int. ICSC/IFAC Symp. Neural Comput.*, Vienna, Austria, 1998, pp. 591–596.

[18] ——, "Learning continuous trajectories in recurrent neural networks with time-dependent weights," *IEEE Trans. Neural Networks*, vol. 10, pp. 741–756, 1999.

[19] J. P. Draye, D. Pavisic, G. Cheron, and G. Libert, "An inhibitory weight initialization improves the speed and quality of recurrent neural network learning," *Neurocomput.*, vol. 16, pp. 207–224, 1997.

[20] R. R. Fedorenko, *Approximate Solutions of Optimal Control Problems* (in Russian). Moscow, Russia: Nauka, 1978.

[21] M. Galicki, "The planning of robotic optimal motions in the presence of obstacles," *Int. J. Robot. Res.*, vol. 17, no. 3, pp. 248–259, 1998.

[22] E. Kochs, K. J. Kalkman, C. Thornton, D. Newton, P. Bischoff, H. Kuppe, J. Abke, E. Konecny, W. Nahm, and G. Stockmanns, "Middle latency auditory evoked responses and electroencephalographic derived variables do not predict movement to noxious stimulation during 1 minimum alveolar anesthetic concentration isoflurane/nitrous oxide anesthesia," *Anesth. Analg.*, vol. 88, pp. 1412–1417, 1999.

[23] F. W. Davies, H. Mantzaridis, G. N. Kenny, and A. C. Fisher, "Middle latency auditory evoked potentials during repeated transitions from consciousness to unconsciousness," *Anaesthesia*, vol. 51, pp. 107–113, 1996.

[24] C. Thornton, "Evoked potentials in anaesthesia," *European J. Anaesthesiology*, vol. 8, pp. 89–107, 1991.